

# Sliding Window Abstraction for Infinite Markov Chains<sup>\*</sup>

Thomas A. Henzinger<sup>1</sup>, Maria Mateescu<sup>1</sup>, and Verena Wolf<sup>1,2</sup>

<sup>1</sup>EPFL, Switzerland

<sup>2</sup>Saarland University, Germany

**Abstract.** We present an on-the-fly abstraction technique for infinite-state continuous-time Markov chains. We consider Markov chains that are specified by a finite set of transition classes. Such models naturally represent biochemical reactions and therefore play an important role in the stochastic modeling of biological systems. We approximate the transient probability distributions at various time instances by solving a sequence of dynamically constructed abstract models, each depending on the previous one. Each abstract model is a finite Markov chain that represents the behavior of the original, infinite chain during a specific time interval. Our approach provides complete information about probability distributions, not just about individual parameters like the mean. The error of each abstraction can be computed, and the precision of the abstraction refined when desired. We implemented the algorithm and demonstrate its usefulness and efficiency on several case studies from systems biology.

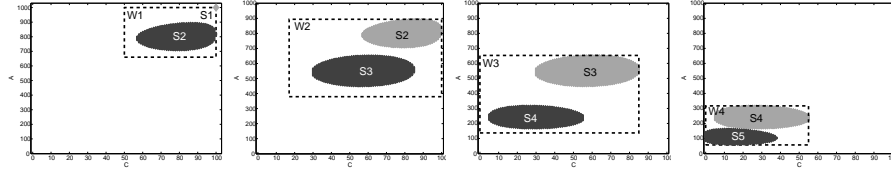
## 1 Introduction

We present a new abstraction technique for infinite-state continuous-time Markov chains (CTMCs) that are specified by a finite set of transition classes. Such models naturally represent biochemical reactions and therefore play an important role in the stochastic modeling of inter- and intracellular processes. A *state* is a vector  $x \in \mathbb{N}_0^n$  whose dimension  $n$  is the number of chemical species, and whose components (the state variables) represent the number of molecules of each species. The analysis of Markov models reveals the biological role of intrinsic noise in gene-network structures, which has received much attention in systems biology [32, 41]. The method of choice has been Monte Carlo simulation. This is because standard numerical solution algorithms for CTMCs do not apply to infinite-state systems, whereas upper bounds for the state variables are rarely known, and even if they are known, the algorithms suffer from a “curse of dimensionality,” i.e., state explosion. The usual measure of interest is the full probability distribution of state variables at various time instances. Repeated simulation, however, while useful to give some information about individual state parameters such as means and variances, is too expensive to obtain more information about probability distributions [?].

In the context of systems biology, the computation of event probabilities is important for several reasons. First, cellular process may decide probabilistically between several possibilities, e.g., in the case of developmental switches [?, 4, 36]. In order to

---

<sup>\*</sup> The research has been partially funded by the Swiss National Science Foundation under grant 205321-111840.



**Fig. 1.** Sliding window method. In each iteration step, the window  $W_j$  captures the set  $S_j$  of states where the significant part of the probability mass is located initially (light gray), the set  $S_{j+1}$  of states that are reached after a time step (dark gray), and the states that are visited in between.

verify, falsify, or refine the mathematical model based on experimental data, the likelihood for each of these possibilities has to be calculated. But also full distributions are of interest, such as the distribution of switching delays [27], the distribution of the time of DNA replication initiation at different origins [?], and the distribution of gene expression products [?]. Finally, many parameter estimation methods require the computation of the posterior distribution because means and variances do not provide enough information to calibrate parameters [?].

If the populations of certain chemical species are large, then the discrete structure of model renders its analysis difficult. However, their effect on the system’s variance may be small, and if this is the case, they can be approximated assuming a continuous deterministic change, and it would be better to use continuous-state approximations such as the Langevin approach [22]. For species with small populations, however, a continuous approximation is not appropriate and other approximation techniques are necessary to reduce the computational effort of the analysis.

We propose a new numerical solution algorithm that approximates the desired probability distributions by solving a sequence of dynamically constructed abstract models. Each abstract model is a finite CTMC that represents the behavior of the original, infinite CTMC during a specific time interval. In each step, we construct geometric boundaries in the original state space which encompass the states where most of the probability mass is located. These boundaries form a “window” that moves through the infinite state space. The state space of each abstract model is formed by the finitely many states inside the window, together with a single absorbing state that represents all of the infinitely many states outside the window. In subsequent time intervals, the window movement follows the direction in which the probability mass moves; see Fig. 1. In each step, the initial conditions are given by a probability vector (whose support is shown in light gray); then an abstract model is constructed (whose state space is depicted by the dashed rectangle), and solved to obtain the next vector (dark gray).

Our approach works well if during each time interval, most of the probability mass is concentrated on a tractable subset of the states. Often real-world systems, such as a biological process, have this property, while, for instance, a random-walk model will become intractable after a certain time period, because the probability mass will be distributed uniformly on the entire infinite state space. We can compute the error of each abstract model, and refine the precision of the abstraction when desired, by enlarging the window (as usual, there is a trade-off between precision and cost of the analysis).

To demonstrate the effectiveness of our approach, we have implemented the algorithm and applied it successfully to several examples from systems biology. The experimental results show that our method allows for an efficient analysis while providing high accuracy. The two most complex examples that we consider are infinite in three dimensions and describe networks of at least seven chemical reactions. It is difficult to find comparison data, because these examples are beyond the scope of what has been handled in other work on solving CTMCs.

**Related work** Various abstraction techniques for Markov chains with *finite* state spaces have been developed during the last years [10, 11, 23, 25]. Infinite-state Markov chains with *discrete time* have been considered in the context of probabilistic lossy-channel systems [1–3, 35] and probabilistic pushdown systems [12–14, 24]. In the infinite-state continuous-time setting, model-checking algorithms for quasi-birth-death processes and Jackson queuing networks have been studied by Remke [37], where the underlying Markov chains are highly structured and represent special cases of CTMCs defined by transition classes. The closest work to ours is the model-checking algorithm for infinite-state CTMCs by Zhang et al. [43]. Depending on the desired precision, their algorithm simply explores the reachable states up to a finite path depth. In contrast, our approach takes into account the direction into which the probability mass moves, and constructs a sequence of abstract models “on-the-fly,” during the verification process. Similar approaches have also been used in the context of biochemical reaction networks. Similar to [43], Munsy et al. [30] explore models up to a specified finite path depth, whereas Burrage et al. [7] consider a finite projection that is doubled if necessary. The latter method, however, requires a priori knowledge about the direction and spread of the probability mass.

## 2 Transition Class Models

Our approach builds on a high-level modeling formalism, called *Transition Class Models* (TCMs), which provides a functional description of structured Markov chains with countably infinite state spaces. TCMs have been used in queuing theory [40] and recently for stochastic models of coupled chemical reactions [38]. We consider a dynamical system with a countable set  $S$  of states.

**Definition 1.** A transition class  $C$  is a triple  $(G, u, \alpha)$  with a guard set  $G \subset S$ , an update function  $u : G \rightarrow S$ , and a rate function  $\alpha : G \rightarrow \mathbb{R}_{>0}$ . A transition class model (TCM) is a pair  $(y, \{C_1, \dots, C_k\})$ , where  $y \in S$  is an initial state and  $\{C_1, \dots, C_k\}$  is a finite set of transition classes.

The guard set  $G$  contains all states  $x$  in which a transition of class  $C$  is possible, and  $u(x)$  is the target state of the transition. Each  $C$ -transition has an associated rate  $\alpha(x)$  that depends on the current state  $x$ .

*Example 1.* We consider a simple birth-death process with  $S = \mathbb{N}_0$ ,  $y = 0$ , and two transition classes  $C_b = (G_b, u_b, \alpha_b)$  and  $C_d = (G_d, u_d, \alpha_d)$ . A birth event increments the value of the state variable by 1 at a constant rate  $\lambda > 0$ , whereas a death event decrements it by 1 at a constant rate  $\mu > 0$ . Formally, for all  $x \in S$ , we define  $G_b = S$ ,  $u_b(x) = x + 1$ ,  $\alpha_b(x) = \lambda$ ,  $G_d = \{x \in S \mid x > 0\}$ ,  $u_d(x) = x - 1$ , and  $\alpha_d(x) = \mu$ .

In practice, we can usually express the sets  $G$  by a finite number of constraints on the state variables of the system, and  $u$  and  $\alpha$  by elementary arithmetic functions.

**Stochastic Semantics.** For a given TCM  $M = (y, \{C_1, \dots, C_k\})$  we derive a continuous-time Markov chain (CTMC)  $(X(t), t \geq 0)$  with state space  $S$ . Let  $i \in \{1, \dots, k\}$  and  $C_i = (G_i, u_i, \alpha_i)$ . We define the probability of a transition of type  $C_i$ , occurring within an infinitesimal time interval  $[t, t + dt)$ , by

$$Pr(X(t + dt) = u_i(x) \mid X(t) = x) = \alpha_i(x) \cdot dt \quad (1)$$

for all  $x \in G_i$ . We call  $\alpha_i(x)$  the *rate* of the transition  $x \rightarrow u_i(x)$ . Note that the transition probability in Eq. 1 does not depend on  $t$  but only on the length of the interval. Moreover, as  $(X(t), t \geq 0)$  possesses the Markov property, the above probability does not depend on the states visited before time  $t$ . Since  $y$  is the initial state of  $M$ , we have  $Pr(X(0) = y) = 1$ , and, for  $x \in S$ , we define the probability that the process is in state  $x$  at time  $t$  by

$$p^{(t)}(x) = Pr(X(t) = x \mid X(0) = y). \quad (2)$$

In the sequel, a matrix description of the transition probabilities is more advantageous. To simplify our presentation, for all  $i$ , we extend the domain of both  $\alpha_i$  and  $u_i$  to  $S$  and set  $\alpha_i(x) = 0$  if  $x \notin G_i$ . Let  $Q : S \times S \rightarrow \mathbb{R}$  be the function that calculates the transition rate of each pair  $(x, x')$  of states with  $x \neq x'$ , that is<sup>1</sup>,

$$Q(x, x') = \sum_{i: u_i(x) = x'} \alpha_i(x). \quad (3)$$

As for the diagonal of the matrix, we set  $Q(x, x) = -\sum_{x' \neq x} Q(x, x')$ . By assuming a fixed enumeration of  $S$ , we can regard  $Q$  as a matrix, called *generator matrix* of  $(X(t), t \geq 0)$ , and describe the evolution of the system by the *Kolmogorov differential equation* [9]

$$\frac{d\mathbf{p}^{(t)}}{dt} = \mathbf{p}^{(t)} \cdot Q. \quad (4)$$

We write  $\mathbf{1}_y$  for the column vector with zeros everywhere except at state  $y$ , where it is one. By  $(\cdot)^T$  we denote matrix transposition. If we assume that the initial condition of the system is  $\mathbf{p}^{(0)} = (\mathbf{1}_y)^T$  and  $(X(t), t \geq 0)$  is a regular Markov chain [9] then Eq. 4 has the unique solution

$$\mathbf{p}^{(t)} = \mathbf{p}^{(0)} \cdot \exp(Qt). \quad (5)$$

Assume that  $|S| < \infty$ . Then  $\exp(Qt) = \sum_{i=0}^{\infty} (Qt)^i / i!$  and, for all  $t \geq 0$ ,  $(\exp(Qt))_{x,z}$  is the probability to reach state  $z$  from  $x$  after  $t$  time units. Analytic solutions for the function  $p^{(t)}$  can only be derived for special cases. If the underlying graph of the CTMC is acyclic, a closed-form expression for  $p^{(t)}(x)$  can be calculated using the recursive scheme of the ACE algorithm [26]. In general, finding the state probabilities as a symbolic function of  $t$  is not possible.

<sup>1</sup> Note that the stochastic semantics ignores self-loops, because they do not have any effect on the probabilities  $p^{(t)}(x)$ .

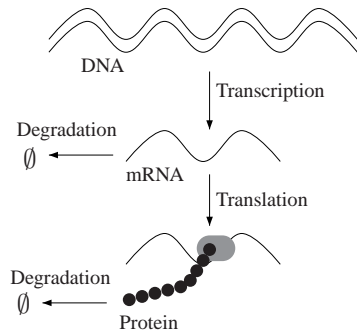
Numerical solutions of the differential equation in Eq. 4 usually exploit the following property. If we split the time interval  $[0, t)$  into  $r$  intervals  $[t_0, t_1), \dots, [t_{r-1}, t_r)$  with  $t_0 < \dots < t_r$ , and  $t_0 = 0, t_r = t$ , then Eq. 5 can be rewritten as

$$\begin{aligned}
\mathbf{p}^{(t_r)} &= \mathbf{p}^{(t_0)} \cdot \exp(Q t_r) \\
&= \mathbf{p}^{(t_0)} \cdot \exp(Q(t_1 - t_0)) \cdot \exp(Q(t_2 - t_1)) \cdot \dots \cdot \exp(Q(t_r - t_{r-1})) \\
&= \mathbf{p}^{(t_1)} \cdot \exp(Q(t_2 - t_1)) \cdot \dots \cdot \exp(Q(t_r - t_{r-1})) \\
&\vdots \\
&= \mathbf{p}^{(t_{r-1})} \cdot \exp(Q(t_r - t_{r-1})).
\end{aligned} \tag{6}$$

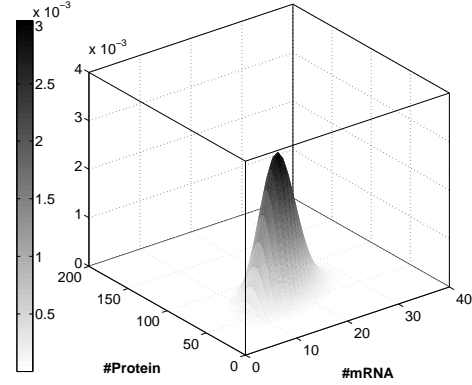
This yields an iterative scheme for the system of differential equations given by Eq. 4. However, numerical solution approaches suffer from the fact that even if upper bounds on the state variables of the system are known, the size of the (truncated) state space is still too large for an efficient solution. We present a way to combat this problem by constructing and analyzing abstract models that approximate the behavior of the infinite-state Markov chain during each interval.

**Biochemical Reaction Networks.** We illustrate our approach by presenting transition class models of cellular chemical systems. According to the theory of *stochastic chemical kinetics* [15], a network of chemical reactions can be modeled as a CTMC. We consider a system of  $n$  different types of molecules and assume that molecules collide randomly and may undergo chemical reactions. If we assume further that the reaction volume (e.g., a cell) has a fixed size and temperature, as well as that the system is well-stirred, a state of the system is given by the numbers of molecules of each type. Hence, the state space  $S$  consists of  $n$ -dimensional vectors  $(x_1, \dots, x_n) \in \mathbb{N}_0^n$ . The different types of reactions are usually specified by means of *stoichiometric equations*. For instance,  $A + B \rightarrow C$  means that if a molecule of type  $A$  hits a molecules of type  $B$ , they may form a complex molecule  $C$ . We call the molecule types that are consumed by a reaction *reactants*; in the above example,  $A$  and  $B$  are reactants. Each reaction type can be described by a transition class, and for a class  $C = (G, u, \alpha)$ , the guard  $G$  contains the states in which enough reactants are available. The update function  $u$  specifies how many molecules of each type are produced and how many are consumed by the reaction. Thus,  $u$  is of the form  $u(x) = x + v$  where  $v \in \mathbb{Z}^n$ . Recall that the probability of a transition of class  $C_i$  occurring in the next infinitesimal time interval of length  $dt$  is  $\alpha_i(x) \cdot dt$ . For a stochastic kinetic analysis,  $\alpha_i(x)$  is the product of a constant and the number of distinct combinations of reactants. This ensures that a chemical reaction is more likely to happen if many reactants are available.

*Example 2.* We consider a simple transition class model for transcription of a gene into messenger RNA (mRNA), and subsequent translation of the latter into proteins [42]. An illustration is given in Fig. 2. This reaction network involves three chemical species, namely, gene, mRNA, and protein. As always only a single copy of the gene exists, a state of the system is uniquely determined by the number of mRNA and protein molecules. Therefore,  $S = \mathbb{N}_0^2$  and a state is a pair  $(x_R, x_P) \in S$ . We assume that initially there are no mRNA molecules and no proteins in the system, i.e.,  $y = (0, 0)$ . Four types of reactions occur in the system. Let  $i \in \{1, \dots, 4\}$  and  $C_i = (G_i, u_i, \alpha_i)$



**Fig. 2.** Transcription of a gene into mRNA and subsequent translation into a protein.



**Fig. 3.** Probability distribution of the gene expression network at  $t = 1000$ .

be the transition class that describes the  $i$ -th reaction type. We first define the guard sets  $G_1, \dots, G_4$  and the update functions  $u_1, \dots, u_4$ .

- Transition class  $C_1$  models gene transcription. If a  $C_1$ -transition occurs, the number of mRNA molecules increases by 1. Thus,  $u_1(x_R, x_P) = (x_R + 1, x_P)$ . This transition class is possible in all states, i.e.,  $G_1 = S$ .
- We represent the translation of mRNA into protein by  $C_2$ . A  $C_2$ -transition is only possible if there is at least one mRNA molecule in the system. We set  $G_2 = \{(x_R, x_P) \in S \mid x_R > 0\}$  and  $u_2(x_R, x_P) = (x_R, x_P + 1)$ . Note that in this case mRNA is a reactant that is not consumed.
- Both mRNA and protein molecules can degrade, which is modeled by  $C_3$  and  $C_4$ . Hence,  $G_3 = G_2$ ,  $G_4 = \{(x_R, x_P) \in S \mid x_P > 0\}$ ,  $u_3(x_R, x_P) = (x_R - 1, x_P)$ , and  $u_4(x_R, x_P) = (x_R, x_P - 1)$ .

Let  $c_1, c_2, c_3, c_4 \in \mathbb{R}_{>0}$  be constants. Gene transcription happens at the constant rate  $\alpha_1(x_R, x_P) = c_1$ , as only one reactant molecule (the gene) is available. The translation rate depends linearly on the number of mRNA molecules. Therefore,  $\alpha_2(x_R, x_P) = c_2 \cdot x_R$ . Finally, for degradation, we set  $\alpha_3(x_R, x_P) = c_3 \cdot x_R$  and  $\alpha_4(x_R, x_P) = c_4 \cdot x_P$ .

Fig. 3 shows the probability distribution  $p^{(t)}$  of the underlying Markov chain after  $t = 1000$  seconds. The parameters are chosen as  $c_1 = 0.05$ ,  $c_2 = 0.0058$ ,  $c_3 = 0.0029$ , and  $c_4 = 10^{-4}$ , where  $c_3$  and  $c_4$  correspond to a half-life of 4 minutes for mRNA and 2 hours for the protein [42]. Most of the probability mass concentrates on the part of the state space where  $5 \leq x_R \leq 30$  and  $25 \leq x_P \leq 110$  and, in a 3D-plot, it forms a steep “hill” whose top represents the states with the highest probability. Note that *every* state in the infinite set  $S$  has a non-zero probability at all time points  $t > 0$ , because the underlying graph of the Markov chain is strongly connected. As time passes, the hill moves through the state space until the distribution reaches its steady-state.

### 3 Abstraction of TCMs

Our approach is based on the observation that a Markov chain describing a certain real-world system often has the following property. After starting with probability 1

Input: TCM $(y, \{C_1, \dots, C_k\})$ , $t_1, \dots, t_r$ with $0 < t_1 < \dots < t_r$ .	
Output: Approximations $\hat{p}^{(t_1)}, \dots, \hat{p}^{(t_r)}$ and error $\epsilon$ .	
1	Initialize $W_0 = \{y\}$ , $p(y) = 1$ , and $\epsilon = 0$ .
2	<b>for</b> $j \in \{1, \dots, r\}$ <b>do</b>
3	Set $h_j = t_j - t_{j-1}$ .
4	Compute $W_j$ depending on $p$ , $C_1, \dots, C_k$ , and $h_j$ .
5	Construct generator $Q_j$ of the abstract model based on $C_1, \dots, C_k$ and $W_j$ .
6	Set $q(x) = \begin{cases} p(x) & \text{if } x \in W_{j-1} \cap W_j, \\ \sum_{x \in W_{j-1} \setminus W_j} p(x) & \text{if } x = x_f, \\ 0 & \text{otherwise.} \end{cases}$
7	Compute $p = q \cdot \exp(Q_j h_j)$ .
8	Set $\hat{p}^{(t_j)}(x) = p(x)$ for $x \in W_j$ and $\hat{p}^{(t_j)}(x) = 0$ otherwise.
9	Set $\epsilon = \epsilon + p(x_f)$ ;
10	<b>end</b>

**Alg.1.** The basic steps of the approximate solution of the Markov chain.

in the initial state  $y$ , the probability mass does not distribute uniformly in  $S$ , such as, for instance, in the case of a random walk. Instead, at each point in time, most of the probability mass distributes among a finite, relatively small number of states. This set of states changes as time progresses, but it never exceeds a certain size. Often, the states with “significant” probability are located at the same part of the state space, as illustrated in Fig. 3.

Let  $(y, \{C_1, \dots, C_k\})$  be a TCM and let  $p^{(t)}$  be the probability distribution of the associated CTMC. We propose an abstraction technique for the computation of  $p^{(t)}$  that proceeds in an iterative fashion. We divide the time interval  $[0, t)$  into  $r$  intervals as in Eq. 6 and approximate  $p^{(t_1)}, \dots, p^{(t_r)}$  by considering a sequence of  $r$  abstractions of the Markov chain under study. Let  $j \in \{1, \dots, r\}$ . In the  $j$ -th step, we construct, on-the-fly, a finite Markov chain for the system behavior during the interval  $[t_{j-1}, t_j)$  from the transition class description. The state space of the  $j$ -th abstract model is the set  $W_j$  of states where most of the probability mass is located during  $[t_{j-1}, t_j)$ . We refer to this set as a *window*. The remaining states are collapsed into a single absorbing state  $x_f$ , i.e., a state that cannot be left.

### 3.1 Algorithm

Alg. 1 describes an iterative method to approximate  $p^{(t_1)}, \dots, p^{(t_r)}$  by vectors  $\hat{p}^{(t_1)}, \dots, \hat{p}^{(t_r)}$ . We start with probability 1 in the initial state  $y$  (line 1). In line 4, we compute the window  $W_j$  such that most of the probability mass remains within  $W_j$  during the next  $h_j$  time units. In line 5, we construct the generator matrix of the abstract model (the finite Markov chain with state space  $W_j \cup \{x_f\}$ ). We define the initial distribution of the abstract model in line 6 and calculate its solution in line 7. The approximation  $\hat{p}^{(t_j)}$  of  $p^{(t_j)}$  is then defined in line 8. Finally, in line 9, we add the approximation error to  $\epsilon$ . A detailed error analysis is given below. Note that after the  $j$ -th loop  $\epsilon = 1 - \sum_{x \in W_j} p(x)$ , that is, in each loop, the probability of being in  $x_f$  may increase. Thus,

$$\sum_{x \in S} \hat{p}^{(t_1)} \leq \dots \leq \sum_{x \in S} \hat{p}^{(t_r)}.$$



The general idea of this abstraction approach is apparent from Fig. 3, but the main difficulty is to find the states that can be neglected in step  $j$  (line 4). In Section 3.2, we explain how to predict the direction and spread of the probability mass during  $[t_{j-1}, t_j]$ .

Let  $\epsilon > 0$ . For an interval  $[t, t+h]$ , we define the size  $m(\epsilon, t, h)$  of the set of significant states as the smallest number for which there exists  $W \subset S$ ,  $|W| = m(\epsilon, t, h)$  such that

$$P(X(t') \in W, t' \in [t, t+h]) \geq 1 - \epsilon. \quad (7)$$

The value  $m(\epsilon, t, h)$  indicates how strongly the probability mass spreads out on  $S$  during  $[t, t+h]$ . Consider, for instance, a random walk on the non-negative integer lattice in the plane that starts in  $(0,0)$  [31]. Between each pair of neighbor states there is a transition with rate 1. For  $h > 0$ , the value  $m(\epsilon, t, h)$  approaches infinity as  $t \rightarrow \infty$ . As opposed to the random walk example, in many systems  $m(\epsilon, t, h)$  is a manageable number of states, even if  $\epsilon$  is small and  $t$  is large (or tends to infinity). Consider, for instance, Ex. 2 and assume that  $h = 500$ ,  $\epsilon = 10^{-6}$ . For each interval  $[t, t+h] \subset [0, \infty)$ ,  $m(\epsilon, t, h)$  does not exceed 20000 states. Alg. 1 works well if  $m(\epsilon, t_{j-1}, h_j)$  is a manageable number of states for all  $j$ . Note that, in particular, cellular usually follow a small number of trends, that is, the quantitative outcomes of a biological experiments can usually be classified within a small number of different categories. Thus, our approach is well suited for TCMs of biological systems.

**Construction of the Abstract Model.** For  $j \in \{1, \dots, r\}$ , let  $W_j$  be such that

$$P(X(h) \in W_j, h \in [t_{j-1}, t_j]) \geq 1 - \epsilon_j \quad (8)$$

where  $\epsilon_j > 0$  is the approximation error of the  $j$ -th step. Note that Eq. 8 implies that  $W_j \cap W_{j+1} \neq \emptyset$ , because the intersection of two successive windows must contain those states that have a high probability at time  $t_j$ . It is far too costly to construct the *smallest* set with this property. Instead, we propose a cheap construction of a set  $W_j$  with a hyper-rectangular shape. We will outline the construction in Section 3.2. The abstract Markov chain of the  $j$ -th step has the finite state space  $W_j \cup \{x_f\}$ , where  $x_f$  represents all states  $x \in S \setminus W_j$ . The transitions of the abstract model are given by the transition classes of the original model except that all transitions of states at the boundary lead to  $x_f$ . Formally, for each class  $C = (G, u, \alpha)$  of the infinite-state Markov chain  $(X(t), t \geq 0)$ , we define  $C' = (G', u', \alpha')$  such that  $G' = G \cap W_j$ ,

$$u'(x) = \begin{cases} u(x) & \text{if } u(x) \in W_j, \\ x_f & \text{otherwise,} \end{cases}$$

and  $\alpha'(x) = \alpha(x)$  for all  $x \in G'$ . Thus, we consider an (extended) subgraph of the one underlying  $(X(t), t \geq 0)$ , with vertexes set  $W_j$ , and all edges leading from  $W_j$  to  $S \setminus W_j$  redirected to the extension  $x_f$ . Note that no transitions are possible from  $x_f$ . We will see that  $x_f$  can be used to calculate the approximation error as it captures the probability mass that leaves  $W_j$ .

**Error Analysis.** Recall that if  $Q$  is the generator matrix of the original Markov chain (cf. Eq. 3),  $\exp(Qh_j)$  is the transition probability matrix for time step  $h_j$ . Let  $Q_j$  be the generator matrix of the abstract Markov chain constructed in the  $j$ -th step (see Alg. 1,



line 5). For  $x, z \in W_j$ , we use the approximation

$$\begin{aligned} (\exp(Qh_j))_{x,z} &= P(X(t_j) = z \mid X(t_{j-1}) = x) \\ &\approx P(X(t_j) = z \wedge X(h) \in W_j, h \in (t_{j-1}, t_j) \mid X(t_{j-1}) = x) \quad (9) \\ &= (\exp(Q_j h_j))_{x,z}. \end{aligned}$$

in line 5 of Alg. 1. Thus, we ignore the probability to reach  $z$  from  $x$  after  $h_j$  time units by leaving  $W_j$  at least once.

For the error analysis, we assume that the vector  $q_j$  of size  $|W_j| + 1$  is such that  $q_j(x) = p^{(t_{j-1})}(x)$  if  $x \in W_j$ . This is true for  $j = 1$  and for  $j > 1$  we replace  $p^{(t_{j-1})}(x)$  by  $\hat{p}^{(t_{j-1})}(x)$  in Alg. 1. In line 7 and 8, we define  $\hat{p}^{(t_j)}(z) = (q_j \cdot \exp(Q_j h_j))_z$  for  $z \in W_j$ . Thus,

$$\begin{aligned} \hat{p}^{(t_j)}(z) &= (q_j \cdot \exp(Q_j h_j))_z \\ &= \sum_{x \in W_j} p^{(t_{j-1})}(x) (\exp(Q_j h_j))_{x,z} \\ &\approx \sum_{x \in W_j} p^{(t_{j-1})}(x) (\exp(Q h_j))_{x,z} \\ &= \sum_{x \in W_j} P(X(t_{j-1}) = x) \cdot P(X(t_j) = z \mid X(t_{j-1}) = x) \\ &\approx \sum_{x \in S} P(X(t_{j-1}) = x) \cdot P(X(t_j) = z \mid X(t_{j-1}) = x) \\ &= P(X(t_j) = z) = p^{(t_j)}(z). \end{aligned} \quad (10)$$

The first approximation is due to Eq. 9. The second approximation comes from the fact that we ignore the probability of not being in  $W_j$  at time  $t_{j-1}$ . In both cases we use an underapproximation. By setting  $\hat{p}^{(t_j)}(z) = 0$  if  $z \notin W_j$ , we obtain  $\hat{p}^{(t_j)}(z) \leq p^{(t_j)}(z)$  for all  $z \in S$ . Overall, we use three approximations, where probability is “lost” namely,

- (a) the probability that is lost due to the approximation given by Eq. 9,
- (b) the probability of not starting in  $W_j$  at time  $t_{j-1}$  (second approximation in Eq. 10),
- (c) the probability of leaving  $W_j$  during  $[t_{j-1}, t_j]$  (which arises due to the approximation  $p^{(t_j)}(z) \approx 0$  if  $z \notin W_j$ ).

It is easy to see that, if the probability of being in  $W_j$  during  $[t_{j-1}, t_j]$  is at least  $1 - \epsilon_j$  (see Eq. 8), then all three errors are at most  $\epsilon_j$ . Thus,  $\|p^{(t_j)} - \hat{p}^{(t_j)}\|_1 \leq \epsilon_j$ . Note that the entry  $p(x_f)$  that is computed in line 7 of Alg. 1 contains all three approximation errors (a), (b), (c). After the termination of the for loop,  $\epsilon$  contains the total approximation error, which is at most  $\epsilon_1 + \dots + \epsilon_r$ .

**Numerical Solution Methods.** For the solution step in line 7 of Alg. 1, we apply a numerical method to compute the matrix exponential. If  $Q_j$  is small then the matrix exponential can be computed efficiently using, for instance, Padé approximation [5, 29]. If the size of  $Q_j$  is large but  $Q_j$  is sparse then iterative methods perform better, such as uniformization [21, 17], approximations in the Krylov subspace [34], or numerical integration [18, 19].

### 3.2 Window Construction

Let us now focus on the construction of the set  $W_j$  in line 7 of Algorithm 1 (see also Eq. 8). Recall that this requires the prediction of the size and location of the probability mass during  $[t_{j-1}, t_j]$ . For arbitrary transition class models, a cheap prediction of

the future behavior of the process is not possible as the transition classes may describe any kind of “unsystematic” behavior. However, many systems have certain linearity properties, which allow for an efficient approximation of the future behavior of the process. Consider a transition class  $C_m = (G_m, u_m, \alpha_m)$ , and assume that the successor  $u_m(x)$  of a state  $x \in G_m$  is computed as  $u_m(x) = x + v_m$ , where  $v_m \in \mathbb{Z}^n$  is a constant change vector. In many applications, a discrete state variable represents the number of instances of a certain system component type, which is incremented or decremented by a small amount. For instance, in the case of biochemical reaction networks,  $v_m \in \{-2, -1, \dots, 2\}^n$ , because a reaction changes the population vectors of the chemical species by an amount of at most two. Any reaction that requires the collision of more than two molecules is usually modeled as a sequence of several reactions. For the rate function  $\alpha_m$ , we assume that the relative difference  $|\alpha_m(x) - \alpha_m(u(x))|/\alpha_m(x)$  is small for all  $x \in G_m$ . This is the case if, for instance,  $\alpha_m$  is linear or at most quadratic in the state variables. According to stochastic chemical kinetics, this assumption is adequate for biochemical reaction networks, because the rate of a reaction is proportional to the number of distinct combinations of reactants. Finally, we assume that the sets  $G_m$  can be represented as intersections of half planes of  $S$ . Again, this assumption holds for biochemical reaction networks, as  $G_m$  refers to the availability of reactant molecules.

The conditions stated above ensure that we can derive geometric boundaries for the window  $W_j$ . More precisely, in line 4 of Alg. 1 we can construct an  $n$ -dimensional hyper-rectangular  $W_j$  such that the left hand of Eq. 8 is close to one. Intuitively, the boundaries of  $W_j$  describe upper and lower bounds on the state variables  $x_1, \dots, x_n$ . Consider, for instance, Fig. 3 and recall that the initial state of the process is  $y = (0, 0)$ . For the rectangle  $W = \{(x_R, x_P) \in S \mid 0 \leq x_R \leq 30, 0 \leq x_P \leq 120\}$ , we have  $P(X(t) \in W, t \in [0, 1000]) \approx 0.99$ .

For the construction of  $W_j$ , we use a technique that considers only the “worst case” behavior of the Markov chain during  $[t_{j-1}, t_j]$  and is therefore cheap compared to the solution of the abstract model. The random variable  $\alpha_m(X(t))$  represents the rate of transition type  $C_m$  at time  $t$ . We can assume that during a small time interval of length  $\Delta$ ,  $\alpha_m(X(t+h))$  is constant, with  $0 \leq h \leq \Delta$ . If  $x$  is the current state then the number of  $C_m$ -transition within the next  $\Delta$  time units is Poisson distributed with parameter  $\alpha_m(x) \cdot \Delta$  [39]. We can approximate this number by the expectation  $\alpha_m(x) \cdot \Delta$  of the Poisson distribution. As we are interested in an upper and lower bound, we additionally consider the standard deviation  $\sqrt{\alpha_m(x) \cdot \Delta}$  of the Poisson distribution. Thus, in the worst case, the number of transitions of type  $C_m$  is

- at least  $\kappa_m^-(x, \Delta) = \max(0, \alpha_m(x) \cdot \Delta - \sqrt{\alpha_m(x) \cdot \Delta})$ ,
- at most  $\kappa_m^+(x, \Delta) = \alpha_m(x) \cdot \Delta + \sqrt{\alpha_m(x) \cdot \Delta}$

Note that if, for instance,  $\alpha_m(x) \cdot \Delta = 1$ , then we have a confidence of 91.97% that the real number of transitions lies in the interval

$$\left[ \alpha_m(x) \cdot \Delta - \sqrt{\alpha_m(x) \cdot \Delta}, \alpha_m(x) \cdot \Delta + \sqrt{\alpha_m(x) \cdot \Delta} \right].$$

Let  $\kappa_m \in \{\kappa_m^+, \kappa_m^-\}$  and  $x^{(0)} = x$ . For  $l = 0, 1, \dots$ , the iteration

$$x^{(l+1)} = x^{(l)} + \sum_{m=1}^k v_m \cdot \kappa_m(x^{(l)}, \Delta) \quad (11)$$

yields worst-case approximations of  $X(t + \Delta), X(t + 2\Delta), \dots$  under the condition that  $X(t) = x$ . Note that  $x^{(l)} \in \mathbb{R}_{\geq 0}^n$ . For functions  $\alpha_m$  that grow extremely fast in the state variables, the iteration may yield bad approximations since it is based on the assumption that the rates are constant during a small interval. In the context of biochemical reaction networks, the linearity properties mentioned above are fulfilled and Eq. 11 yields adequate approximations. The bounds  $b_d^+(x)$  and  $b_d^-(x)$  for dimension  $d \in \{1, \dots, n\}$  are given by the minimal and maximal values during the iteration. More precisely,  $b_d^+(x) = \lceil \max_l x_d^{(l)} \rceil$  and  $b_d^-(x) = \lfloor \min_l x_d^{(l)} \rfloor$ , where  $x^{(l)} = (x_1^{(l)}, \dots, x_n^{(l)})$ .

In order to construct  $W_j$ , we do not consider *all* combinations  $\{\kappa_1^+, \kappa_1^-\} \times \dots \times \{\kappa_k^+, \kappa_k^-\}$  in Eq. 11. We choose only those combinations that do not treat preferentially transition types leading to opposite directions in the state space. Consider, for instance, Ex. 2 with  $x = (5, 50)$  and  $\Delta = 10$ . If we assume that more reactions of type  $C_1$  and  $C_2$  happen (than on average) and fewer of  $C_3$  and  $C_4$ , we get  $\kappa_1^+(x, \Delta) = c_1 \cdot 10 + \sqrt{c_1 \cdot 10} = 1.2$ ,  $\kappa_2^+(x, \Delta) = c_2 \cdot 10 \cdot 5 + \sqrt{c_2 \cdot 10 \cdot 5} = 0.83$ ,  $\kappa_3^-(x, \Delta) = \max(0, c_3 \cdot 10 \cdot 5 - \sqrt{c_3 \cdot 10 \cdot 5}) = 0$ ,  $\kappa_4^-(x, \Delta) = \max(0, c_4 \cdot 10 \cdot 50 - \sqrt{c_4 \cdot 10 \cdot 50}) = 0$ . This means that the number of protein and mRNA molecules increases and  $x^{(1)} = (6.2, 50.83)$ . We do not consider the combinations that contain both  $\kappa_1^+$  and  $\kappa_3^-$ . As  $C_1$  equates  $C_3$  and vice versa, these combinations do not result in extreme values of the state variables. For each dimension, we can identify two combinations that yield minimal and maximal values by examining the vector field of the transition classes. We refer to a chosen combination as a *branch* and fix for each transition class  $C_m$  a choice  $\kappa_m = \kappa_m^+$  or  $\kappa_m = \kappa_m^-$  for all  $l$ .

For the construction of  $W_j$ , we first need to define the significant set of states at time  $t_{j-1}$ . A very precise method would require sorting of the vector  $\hat{p}^{(t_{j-1})}(x)$ , which we find far too expensive. Therefore, we opt for a simpler solution where we define the set  $S_j = \{x \in S \mid \hat{p}^{(t_{j-1})}(x) > \delta\}$  of states significant at time  $t_{j-1}$ . Here,  $\delta > 0$  is a small constant that is several orders of magnitude smaller than the desired precision. For our experimental results, we used  $\delta = 10^{-10}$  and decreased this value during the iteration if  $\sum_{x \notin S_j} \hat{p}^{(t_{j-1})}(x)$  exceeded our desired precision. For each branch, we carry out the iteration in Eq. 11 for  $\lceil h_j / \Delta \rceil$  steps with 10 different initial states randomly chosen from  $S_j$ . This yields a cheap approximation of the behavior of the process during the interval  $[0, h_j)$ . For dimension  $d$ , let  $b_d^+$  and  $b_d^-$  denote the bounds that we obtain by merging the bounds of each branch and each randomly chosen state. We set

$$W_j = S_j \cup \{x = (x_1, \dots, x_n) \in S \mid b_d^- \leq x_d \leq b_d^+, 1 \leq d \leq n\}.$$

We choose the time steps  $\Delta$  in the order of the expected residence time of the current state such that the assumption of  $\alpha_m(X(t))$  being constant is reasonable.

The boundaries of the window become rough if  $h_j$  is large. Therefore, for the experimental results in Section 4, we choose  $h_j$  dynamically. During the iterative computation of the bounds  $b_d^+$  and  $b_d^-$ , we compute the size of the current window  $W_j$ . We stop the iteration if  $|W_j|$  exceeds twice the size of  $S_j$  but not before  $W_j$  has reached a minimal size of 5000 states. By doing so, we induce a sliding of the window, which is forced to move from its previous location. It is, of course, always possible to choose a smaller value for  $h_j$  if the distribution at a specific time instant  $t < t_{j-1} + h_j$  is of interest.

**Precision.** If the approximation error  $\epsilon$  in Alg. 1 exceeds the desired error threshold, the window construction can be repeated using a larger window  $W_j$ . This may hap-

pen if the confidence of the estimated interval  $[\kappa_m^-(x, \Delta), \kappa_m^+(x, \Delta)]$  for the number of transitions of type  $m$  is not large enough. In this case, the approximation  $\hat{p}^{(t_j)}$  can be used to determine where to expand  $W_j$ . Several heuristics for the window expansion are possible. The smooth distribution of the probability mass, however, suggests to expand only those boundaries of  $W_j$  where states with a high probability are located.

## 4 Experimental Results

We implemented Alg. 1 in C++ and run experiments on a 3.16 GHz Intel Linux PC with 6 GB of RAM. We consider various examples and present our results in Table 1. For the parameters, we used values from literature. We provide a detailed description of the parameters and branches that we used for the gene expression example (cf. Ex. 2). For the parameter details of the remaining examples, we refer to [?].

First we applied our method to finite examples and those we found analyzed in the literature (Examples 1-3). We then considered significantly larger examples (Examples 4-6). The examples studied in [43, 30] are much simpler than the ones presented here. These approaches explore the reachable states up to a certain depth, which yields large spaces that contain many states with a very small probability. For instance, in the case of Ex. 2 the path depth is proportional to the expected number of protein molecules (see also Fig. 3). The states reachable within this fixed number of steps always include states with a high number of mRNA molecules although their probability is very small. In contrast, with our method, we achieve a similar accuracy while using windows that are much smaller. We are therefore able to handle more complex examples.

The enzyme reaction is a prototype of a stiff model and has 5151 states. The crystallization example is also finite but has 5499981 reachable states. To the best of our knowledge, the crystallization, the gene expression, and the virus example have not yet been solved numerically by others. Goutsias model and the enzyme reaction have been considered by Burrage et al. [7], but as already stated, their method requires additional knowledge about the direction and spread of the probability mass.

For the gene expression example, we choose  $y = (0, 0)$ , a time horizon of  $t = 10000$ , and rate constants  $c_1 = 0.05$ ,  $c_2 = 0.0058$ ,  $c_3 = 0.0029$ , and  $c_4 = 10^{-4}$ , where  $c_3$  and  $c_4$  correspond to a half-life of 4 minutes for mRNA and 2 hours for the protein [42]. We use four branches for the iteration given by Eq. 11. We maximize the number of mRNA molecules by choosing  $\kappa_1^+$  and  $\kappa_3^-$  and minimize it with  $\kappa_1^-$  and  $\kappa_3^+$ . Transition classes  $C_2$  and  $C_4$  are irrelevant for this species. We maximize the number of proteins by choosing  $\kappa_1^+$ ,  $\kappa_2^+$ ,  $\kappa_3^-$ , and  $\kappa_4^-$ . The number of proteins becomes minimal with  $\kappa_1^-$ ,  $\kappa_2^-$ ,  $\kappa_3^+$ , and  $\kappa_4^+$ .

In Table 1, the column *ref.* refers to the literature where the example has been presented. Column *#dim* lists the number of dimensions of the set of reachable states. Note that this is not equal to the number of chemical species since certain conservation laws may hold<sup>2</sup> or the copy number of some species can never exceed a small value. Column *#classes* refers to the number of transition classes, and column *infinite* indicates whether the model has an infinite number of states or not. In column *mean*  $|W_j|$  we

<sup>2</sup> For instance, in the case of complex formation the number of complex molecules is uniquely determined by the initial populations and the remaining number of complex components.

name	ref.	#dim	#classes	infinite	mean $ W_j $	running time unif.	running time Krylov	%constr.	error	steps
enzyme reaction	[7]	2	3	no	1437	15 sec	6 sec	6	$1 \times 10^{-5}$	4
crystallization	[20]	2	2	no	47229	8907 sec	4289 sec	30	$2 \times 10^{-7}$	5175
protein synthesis	[43]	1	4	yes	673	3 sec	2 sec	< 1	$2 \times 10^{-7}$	2
gene expression	[42]	2	4	yes	32404	107 sec	98 sec	39	$2 \times 10^{-5}$	87
Goutsias model	[7]	3	10	yes	538815	15943 sec	8412 sec	15	$7.6 \times 10^{-5}$	101
Virus model	[6]	3	7	yes	1092441	27850 sec	11814 sec	9	$6 \times 10^{-6}$	51

**Table 1.** Experimental results of the sliding window method.

list the mean size of the windows that we considered during the iteration. The running times of the sliding window method are given in the column *running times* and *%constr.* refers to the percentage of time used for the construction of the window boundaries and the generator matrices  $Q_j$ . For the computation of the matrix exponential (compare line 7 of Alg. 1) we used both the uniformization method and the Krylov subspace method. The column *error* in Table 1 refers to the total approximation error of our method. The column *steps* in Table 1 gives the number  $r$  of steps in Alg. 1. For each example, the method yields accurate results. We never had to recompute  $\hat{p}^{(t_j)}$  because too much probability was lost. The numerical solution of the abstract models takes most of the running time whereas the window construction takes less than 40% of the running time. Since the memory requirements of the sliding window method are not excessive as it is the case for other methods, we are able to numerically approximate the solution of complex models that have not been solved before.

## 5 Conclusion

The sliding window method is a new approach to analyze infinite-state continuous-time Markov chains. The method applies in particular to Markov chains that arise from networks of biochemical reactions. It is therefore a promising approach for the analysis of cellular stochasticity, which has become increasingly important in recent years.

We approximate the probability distributions of the infinite Markov chain at various time instances by solving a sequence of dynamically constructed finite Markov chains. The abstract models can be solved with any existing numerical algorithm for finite Markov chains. Moreover, it is possible to combine our approach with other techniques, such as time scale separation methods [20, 33, 8].

We demonstrated the effectiveness of our method with a number of experiments. The results show that we can solve more complex systems than previous approaches in acceptable time.

As further enhancements, we plan to develop a steady-state detection mechanism, which allows us to compute the steady-state distribution if the location and size of the window becomes stable. Moreover, we plan to investigate a splitting of the windows, which will be particularly useful for multistable systems.

## References

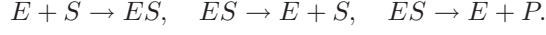
1. P. Abdulla, C. Baier, S. Iyer, and B. Jonsson. Reasoning about probabilistic lossy channel systems. In *Proc. CONCUR'00*, LNCS, pages 320–333. Springer, 2000.
2. P. Abdulla, N. Bertrand, A. Rabinovich, and P. Schnoebelen. Verification of probabilistic systems with faulty communication. *Inf. Comput.*, 202(2):141–165, 2005.
3. P. Abdulla, N. B. Henda, and R. Mayr. Verifying infinite Markov chains with a finite attractor or the global coarseness property. In *Proc. LICS '05*, pages 127–136. IEEE Computer Society, 2005.
4. A. Arkin, J. Ross, and H. H. McAdams. Stochastic kinetic analysis of developmental pathway bifurcation in phage  $\lambda$ -infected escherichia coli cells. *Genetics*, 149:1633–1648, 1998.
5. G.A. Baker. *The essentials of Padé approximants*. Academic Press, N. Y., 1975.
6. R. De Boer. *Theoretical Fysiology*. Online Lecture Notes, 2006.
7. K. Burrage, M. Hegland, F. Macnamara, and B. Sidje. A Krylov-based finite state projection algorithm for solving the chemical master equation arising in the discrete modelling of biological systems. In *Proc. of the Markov 150th Anniversary Conference*, pages 21–38. Boson Books, 2006.
8. H. Busch, W. Sandmann, and V. Wolf. A numerical aggregation algorithm for the enzyme-catalyzed substrate conversion. In *Proc. CMSB*, volume 4210 of LNCS, pages 298–311. Springer, 2006.
9. E. Çinlar. *Introduction to Stochastic Processes*. Prentice-Hall, 1975.
10. P. D'Argenio, B. Jeannet, H. Jensen, and K. Larsen. Reachability analysis of probabilistic systems by successive refinements. In *Proc. PAPM-PROBMIV'01*, pages 39–56, 2001.
11. L. de Alfaro and R. Pritam. Magnifying-lens abstraction for Markov decision processes. In *Proc. CAV*, volume 4590 of LNCS, pages 325–338. Springer, 2007.
12. J. Esparza and K. Etessami. Verifying probabilistic procedural programs. In *Proc. FSTTCS'04*, volume 3328 of LNCS, pages 16–31. Springer, 2005.
13. J. Esparza, A. Kucera, and R. Mayr. Model checking probabilistic pushdown automata. In *Proc. LICS '04*, pages 12–21. IEEE Computer Society, 2004.
14. K. Etessami and M. Yannakakis. Algorithmic verification of recursive probabilistic state machines. In *Proc. TACAS'05*, LNCS, pages 253–270. Springer, 2005.
15. D. T. Gillespie. *Markov Processes*. Academic Press, N. Y., 1992.
16. J. Goutsias. Quasiequilibrium approximation of fast reaction kinetics in stochastic biochemical systems. *J. Chem. Phys.*, 122(18):184102, 2005.
17. D. Gross and D. Miller. The randomization technique as a modeling tool and solution procedure for transient Markov processes. *Operations Research*, 32(2):926–944, 1984.
18. E. Hairer, S. Norsett, and G. Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems*. Springer, 2008.
19. E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems*. Springer, 2004.
20. E. Haseltine and J. Rawlings. Approximate simulation of coupled fast and slow reactions for stochastic chemical kinetics. *The Journal of Chemical Physics*, 117(15):6959–6969, 2002.
21. A. Jensen. Markoff chains as an aid in the study of Markoff processes. *Skandinavisk Aktuarietidskrift*, 36:87–91, 1953.
22. N. G. van Kampen. *Stochastic Processes in Physics and Chemistry*. Elsevier, 3rd edition, 2007.
23. J.-P. Katoen, D. Klink, M. Leucker, and V. Wolf. Three-valued abstraction for continuous-time Markov chains. In *Proc. CAV*, volume 4590 of LNCS, pages 316–329. Springer, 2007.
24. A. Kucera. Methods for quantitative analysis of probabilistic pushdown automata. *Electr. Notes Theor. Comput. Sci.*, 149(1):3–15, 2006.

25. M. Kwiatkowska, G. Norman, and D. Parker. Game-based abstraction for Markov decision processes. In *QEST*, pages 157–166. IEEE CS Press, 2006.
26. R. A. Marie, A. L. Reibman, and K. S. Trivedi. Transient analysis of acyclic Markov chains. *Perform. Eval.*, 7(3):175–194, 1987.
27. H. H. McAdams and A. Arkin. Stochastic mechanisms in gene expression. *Proceedings of the National Academy of Science, USA*, 94:814–819, 1997.
28. D. Milutinović and R. De Boer. Process noise: An explanation for the fluctuations in the immune response during acute viral infection. *Biophysical Journal*, 92:3358–3367, 2007.
29. C. Moler and C. Van Loan. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM Review*, 45(1):3–49, 2003.
30. B. Munsky and M. Khammash. The finite state projection algorithm for the solution of the chemical master equation. *J. Chem. Phys.*, 124:044144, 2006.
31. J. Norris. *Markov Chains*. Cambridge University Press, 1. edition, 1999.
32. J. Paulsson. Summing up the noise in gene networks. *Nature*, 427(6973):415–418, 2004.
33. S. Peles, B. Munsky, and M. Khammash. Reduction and solution of the chemical master equation using time scale separation and finite state projection. *J. Chem. Phys.*, 125:204104, 2006.
34. B. Philippe and R. Sidje. Transient solutions of Markov processes by Krylov subspaces. In *Proc. International Workshop on the Numerical Solution of Markov Chains*, pages 95–119. Kluwer Academic Publishers, 1995.
35. A. Rabinovich. Quantitative analysis of probabilistic lossy channel systems. *Inf. Comput.*, 204(5):713–740, 2006.
36. C. Rao, D. Wolf, and A. Arkin. Control, exploitation and tolerance of intracellular noise. *Nature*, 420(6912):231–237, 2002.
37. A. Remke. *Model Checking Structured Infinite Markov Chains*. PhD thesis, 2008.
38. W. Sandmann and V. Wolf. A computational stochastic modeling formalism for biological networks. In *Enformatika Transactions on Engineering, Computing and Technology*, volume 14, pages 132–137, 2006.
39. W. J. Stewart. *Introduction to the Numerical Solution of Markov Chains*. Princeton University Press, 1995.
40. C. Strelen. Approximate disaggregation-aggregation solutions for general queueing networks. In *Society for Computer Simulation*, pages 773–778, 1997.
41. P. S. Swain, M. B. Elowitz, and E. D. Siggia. Intrinsic and extrinsic contributions to stochasticity in gene expression. *Proceedings of the National Academy of Science, USA*, 99(20):12795–12800, 2002.
42. M. Thattai and A. van Oudenaarden. Intrinsic noise in gene regulatory networks. *PNAS, USA*, 98(15):8614–8619, July 2001.
43. L. Zhang, H. Hermanns, E. Moritz Hahn, and B. Wachter. Time-bounded model checking of infinite-state continuous-time Markov chains. In *ACSD*, 2008. China.



## A Appendix

1. **Enzyme Reaction.** We describe an *enzyme-catalyzed substrate conversion* by the three reactions



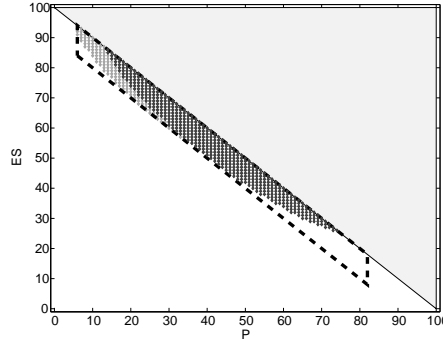
This network involves four chemical species, namely, enzyme ( $E$ ), substrate ( $S$ ), complex ( $ES$ ), and product ( $P$ ) molecules. The transition classes  $C_1, C_2$ , and  $C_3$  are such that for  $(x_1, x_2, x_3, x_4) \in \mathbb{N}^4$ ,  $C_i = (G_i, u_i, \alpha_i)$  for  $1 \leq i \leq 3$ ,

$$\begin{aligned} G_1 &= \{(x_1, x_2, x_3, x_4) \in \mathbb{N}^4 \mid x_1, x_2 > 0\}, \\ u_1(x_1, x_2, x_3, x_4) &= (x_1 - 1, x_2 - 1, x_3 + 1, x_4), \\ \alpha_1(x_1, x_2, x_3, x_4) &= c_1 x_1 x_2, \\ G_2 &= \{(x_1, x_2, x_3, x_4) \in \mathbb{N}^4 \mid x_3 > 0\}, \\ u_2(x_1, x_2, x_3, x_4) &= (x_1 + 1, x_2 + 1, x_3 - 1, x_4), \\ \alpha_2(x_1, x_2, x_3, x_4) &= c_2 x_3, \\ G_3 &= G_2, \\ u_3(x_1, x_2, x_3, x_4) &= (x_1 + 1, x_2, x_3 - 1, x_4 + 1), \\ \alpha_3(x_1, x_2, x_3, x_4) &= c_3 x_3. \end{aligned}$$

The set of states reachable from the initial state  $y = (y_1, y_2, y_3, y_4)$  is finite because of the conservation laws  $y_1 = x_1 + x_3$  and  $y_2 = x_2 + x_3 + x_4$ , where we assume that  $y_3 = y_4 = 0$ . Note that a state is uniquely determined by the initial conditions and its last two entries. Thus, the set of reachable states is two-dimensional. For our experimental results, we chose the same parameters as in [7], that is, initial state  $y = (1000, 100, 0, 0)$ , time horizon  $t = 70$ , and rate constants  $c_1 = c_2 = 1$  and  $c_3 = 0.1$ . With these parameter, the number of states reachable from  $y$  is 5151.

We consider four branches for the iteration in Eq. 11 in order to determine upper and lower bounds on the state variables.

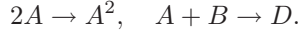
- (a) To obtain an estimate for the maximal number of complex molecules (and a minimum for the enzyme population), we enforce more transitions of type  $C_1$  than on average ( $\kappa_1 = \kappa_1^+$ ), and fewer of types  $C_2$  and  $C_3$  ( $\kappa_3 = \kappa_3^-$  and  $\kappa_2 = \kappa_2^-$ ).
- (b) By considering fewer transitions of type  $C_1$  ( $\kappa_1 = \kappa_1^-$ ), and more of types  $C_2$  and  $C_3$  ( $\kappa_3 = \kappa_3^+$  and  $\kappa_2 = \kappa_2^+$ ) the complex population becomes minimal (and the enzyme population maximal).
- (c) An estimate for the minimal number of type  $P$  molecules (and the maximal number of type  $S$  molecules) is obtained by enforcing more transitions of type  $C_2$  ( $\kappa_2 = \kappa_2^+$ ), and fewer of types  $C_1$  and  $C_3$  ( $\kappa_1 = \kappa_1^-$  and  $\kappa_3 = \kappa_3^-$ ).
- (d) Finally, more transitions of types  $C_1$  and  $C_3$  ( $\kappa_1 = \kappa_1^+$  and  $\kappa_3 = \kappa_3^+$ ), and fewer of type  $C_2$  ( $\kappa_2 = \kappa_2^-$ ) gives a maximal increase of the number of product molecules (and minimizes the number of substrate molecules).



**Fig. 4.** For the enzyme reaction, the window has the shape of a parallelogram.

As illustrated in Fig. 4, the window has a parallelogram shape, because in the two-dimensional setting the bounds on the four state variables and the conservation laws imply upper and lower bounds for the variable  $x_3 + x_4$ . For the experimental results in Table 1, the time horizon  $[0, 70]$  is divided into intervals of length 0.78, 32.11, 12.36, and 24.73. The corresponding window sizes are 5015, 591, 90, and 55. The window size decreases, since more and more probability mass is contained in the absorbing state  $(1000, 0, 0, 100)$ . If the iteration given by Eq. 11 reaches an absorbing state, the window cannot be expanded further and the iteration terminates.

2. **Crystallization.** We consider a *crystallization example* that involves the chemical species  $A$ ,  $A^2$ ,  $B$ , and  $D$ . The two possible reaction types are given by



Let  $S = \mathbb{N}^4$  and  $(x_1, x_2, x_3, x_4) \in S$ . The transition class that corresponds to the first reaction is defined as  $C_1 = (G_1, u_1, \alpha_1)$ , where  $G_1 = \{(x_1, x_2, x_3, x_4) \in S \mid x_1 \geq 2\}$  (there must be at least two molecules of type  $A$ ), and  $u_1(x_1, x_2, x_3, x_4) = (x_1 - 2, x_2 + 1, x_3, x_4)$ . The rate function  $\alpha_1$  takes into account that there are  $\binom{x_1}{2}$  distinct combinations of reactants for the first reaction if  $x_1$  is the number of molecules of type  $A$ . We define  $\alpha_1(x_1, x_2, x_3, x_4) = c_1 \binom{x_1}{2} = c_1 x_1(x_1 - 1)/2$ , where  $c_1 > 0$  is a constant. For the second reaction we define  $C_2 = (G_2, u_2, \alpha_2)$  with  $G_2 = \{(x_1, x_2, x_3, x_4) \in S \mid x_1, x_3 > 0\}$  (there must be at least one  $A$  and one  $B$  molecule), and  $u_2(x_1, x_2, x_3, x_4) = (x_1 - 1, x_2, x_3 - 1, x_4 + 1)$ . Let  $c_2 > 0$  be a constant and define  $\alpha_2(x_1, x_2, x_3, x_4) = c_2 x_1 x_3$ .

Note that for a given initial state  $y = (y_1, y_2, y_3, y_4)$  the finite set of reachable states is given by

$$\{(x_1, x_2, x_3, x_4) \in S \mid y_1 = x_1 + 2 \cdot x_2 + x_4, y_3 = x_3 + x_4\},$$

where we assume for simplicity that  $y_2 = y_4 = 0$ . Moreover, a state is uniquely determined by its first and third entry. Thus, the set of reachable states is two-

dimensional. For our experimental results, we chose the same parameters as in [20], that is, initial state  $y = (10^6, 0, 10, 0)$ , time horizon of  $t = 100$ , and the rate constants  $c_1 = c_2 = 10^{-7}$ . With these parameters, the number of reachable states is 5499981.

The first and third state variables can only decrease, and in order to apply the iteration in Eq. 11 we choose two branches: one for the minimal number of  $A$  molecules, and one for the minimal number of  $B$  molecules. For the population of  $A$ , in the worst case we have more transitions (than on average) of both types,  $C_1$  and  $C_2$ . The number of  $B$  molecules decreases strongly in the case of fewer  $C_1$  transitions and more  $C_2$  transitions. Hence, for the iteration in Eq. 11 we choose  $\kappa_1 = \kappa_1^+$  and  $\kappa_2 = \kappa_2^+$  in the former case, and  $\kappa_1 = \kappa_1^-$  and  $\kappa_2 = \kappa_2^+$  in the latter case.

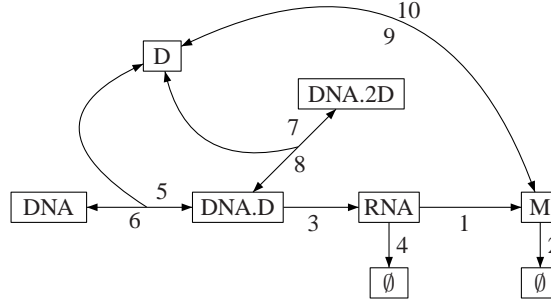
The system is, as the previous example, highly stiff, that is, the reactions occur on very different time scales. Moreover, the computational complexity is huge since all molecules of type  $A$  have to react until an absorbing state is reached. Therefore the number of iteration steps is large and recomputations may occur.

3. **Protein Synthesis.** We describe a simple model of protein synthesis by a transition class model  $(y, \{C_1, C_2, C_3, C_4\})$  with  $y = (0, 0) \in \mathbb{N}^2$ . The first state variable represents the state of a gene for which we only distinguish between an active (value 1) and an inactive state (value 0). If the gene is active, protein molecules are produced. The second state variable equals the current number of proteins. Thus, the set of reachable states is given by  $S = \{0, 1\} \times \mathbb{N}$ . The first transition class represents the activation of the gene, i.e.,  $C_1 = (G_1, u_1, \alpha_1)$ ,  $G_1 = \{(x_1, x_2) \in S \mid x_1 = 0\}$ ,  $u_1(x_1, x_2) = (1, x_2)$ ,  $\alpha_1(x_1, x_2) = c_1$ . Its deactivation is modeled as  $C_2 = (G_2, u_2, \alpha_2)$ ,  $G_2 = \{(x_1, x_2) \in S \mid x_1 = 1\}$ ,  $u_2(x_1, x_2) = (0, x_2)$ ,  $\alpha_2(x_1, x_2) = c_2$ . We describe protein synthesis by  $C_3 = (G_3, u_3, \alpha_3)$ ,  $G_3 = \{(x_1, x_2) \in S \mid x_1 = 1\}$ ,  $u_3(x_1, x_2) = (x_1, x_2 + 1)$ ,  $\alpha_3(x_1, x_2) = c_3$ . Transition class  $C_4 = (G_4, u_4, \alpha_4)$  models the degradation of protein molecules. We define  $G_4 = \{(x_1, x_2) \in S \mid x_2 > 0\}$ ,  $u_4(x_1, x_2) = (x_1, x_2 - 1)$ ,  $\alpha_4(x_1, x_2) = x_2 c_4$ . It is important to note that the system is infinite in only one dimension.

We maximize the number of proteins by choosing  $\kappa_1 = \kappa_1^+$ ,  $\kappa_2 = \kappa_2^-$ ,  $\kappa_3 = \kappa_3^+$ , and  $\kappa_4 = \kappa_4^-$  in Eq. 11. The opposite branch ( $\kappa_1 = \kappa_1^-$ ,  $\kappa_2 = \kappa_2^+$ ,  $\kappa_3 = \kappa_3^-$ , and  $\kappa_4 = \kappa_4^+$ ) computes a minimal bound for the number of proteins.

For the reaction rate constants, we choose the same numbers as in [43], i.e.,  $c_1 = 1$ ,  $c_2 = 5$ ,  $c_3 = 1$ ,  $c_4 = 0.02$ . With the approach in [43], results for a time horizon of at most  $t = 45$  were obtained. With the sliding window method, we are able to consider the system during an arbitrary time horizon. We chose  $t = 1000$  for the results in Table 1.

4. **Gene Expression.** Details about the parameters of the gene expression example are given in Section 4. In Figure 2, we illustrate the probability distribution of the example at  $t = 1000$ .
5. **Goutsias Model.** In [16], Goutsias defines a model for the transcription regulation of a repressor protein in bacteriophage  $\lambda$ . This protein is responsible for maintaining lysogeny of the  $\lambda$  virus in *E. coli*. The model involves 6 different species and 10 reactions. Thus, a state is a vector  $x = (x_1, x_2, x_3, x_4, x_5, x_6) \in \mathbb{N}_0^6$ . The transition classes  $C_i = (G_i, u_i, \alpha_i)$ ,  $1 \leq i \leq 10$  are given as follows.



**Fig. 5.** Dependency graph of Goutsias model.

- Production of proteins:  $G_1 = \{x \in \mathbb{N}_0^6 \mid x_3 > 0\}$ ,  $u_1(x) = (x_1 + 1, x_2, x_3, x_4, x_5, x_6)$ ,  $\alpha_1(x) = c_1 x_3$ .
- Degradation of proteins:  $G_2 = \{x \in \mathbb{N}_0^6 \mid x_1 > 0\}$ ,  $u_2(x) = (x_1 - 1, x_2, x_3, x_4, x_5, x_6)$ ,  $\alpha_2(x) = c_2 x_1$ .
- Production of mRNA:  $G_3 = \{x \in \mathbb{N}_0^6 \mid x_5 > 0\}$ ,  $u_3(x) = (x_1, x_2, x_3 + 1, x_4, x_5, x_6)$ ,  $\alpha_3(x) = c_3 x_5$ .
- Degradation of mRNA:  $G_4 = \{x \in \mathbb{N}_0^6 \mid x_3 > 0\}$ ,  $u_4(x) = (x_1, x_2, x_3 - 1, x_4, x_5, x_6)$ ,  $\alpha_4(x) = c_4 x_3$ .
- First dimer binding at operator site:  $G_5 = \{x \in \mathbb{N}_0^6 \mid x_2, x_4 > 0\}$ ,  $u_5(x) = (x_1, x_2 - 1, x_3, x_4 - 1, x_5 + 1, x_6)$ ,  $\alpha_5(x) = c_5 x_2 x_4$ .
- First dimer unbinding:  $G_6 = \{x \in \mathbb{N}_0^6 \mid x_5 > 0\}$ ,  $u_6(x) = (x_1, x_2 + 1, x_3, x_4 + 1, x_5 - 1, x_6)$ ,  $\alpha_6(x) = c_6 x_5$ .
- Second dimer binding at operator site:  $G_7 = \{x \in \mathbb{N}_0^6 \mid x_2, x_5 > 0\}$ ,  $u_7(x) = (x_1, x_2 - 1, x_3, x_4, x_5 - 1, x_6 + 1)$ ,  $\alpha_7(x) = c_7 x_2 x_5$ .
- Second dimer unbinding:  $G_8 = \{x \in \mathbb{N}_0^6 \mid x_6 > 0\}$ ,  $u_8(x) = (x_1, x_2 + 1, x_3, x_4, x_5 + 1, x_6 - 1)$ ,  $\alpha_8(x) = c_8 x_6$ .
- Dimerization:  $G_9 = \{x \in \mathbb{N}_0^6 \mid x_1 > 1\}$ ,  $u_9(x) = (x_1 - 2, x_2 + 1, x_3, x_4, x_5, x_6)$ ,  $\alpha_9(x) = c_9 x_1(x_1 - 1)/2$ .
- Dissociation into monomers:  $G_{10} = \{x \in \mathbb{N}_0^6 \mid x_2 > 0\}$ ,  $u_{10}(x) = (x_1 + 2, x_2 - 1, x_3, x_4, x_5, x_6)$ ,  $\alpha_{10}(x) = c_{10} x_2$ .

We use the following values for  $c_1, \dots, c_{10}$  (see [16, 7]):  $c_1 = 0.043$ ,  $c_2 = 0.0007$ ,  $c_3 = 0.0715$ ,  $c_4 = 0.0039$ ,  $c_5 = 1.992647 \times 10^{-2}$ ,  $c_6 = 0.4791$ ,  $c_7 = 1.992647 \times 10^{-4}$ ,  $c_8 = 8.765 \times 10^{-12}$ ,  $c_9 = 8.30269 \times 10^{-2}$ , and  $c_{10} = 0.5$ . The initial state of the system is given by  $y = (2, 6, 0, 2, 0, 0)$ .

It is important to note that the last three state variables can only take few different values. Therefore the model is infinite in 3 dimensions only. Figure 5 shows the dependency graph of the model from which the branches can be derived. The nodes of the hypergraph are the chemical species and the edges connect reactants and products of the reactions, respectively. Below, we list the branches for upper bounds on the state variables. Lower bounds are obtained if the opposite combination is considered, respectively.

- We maximize the number of RNA molecules by choosing the combination  $\kappa_1^-, \kappa_2^-, \kappa_3^+, \kappa_4^-, \kappa_5^+, \kappa_6^-, \kappa_7^+, \kappa_8^+, \kappa_9^+, \kappa_{10}^-$ .

- We maximize the number of monomer molecules by choosing the combination  $\kappa_1^+, \kappa_2^-, \kappa_3^+, \kappa_4^-, \kappa_5^+, \kappa_6^-, \kappa_7^-, \kappa_8^+, \kappa_9^-, \kappa_{10}^+$ .
- We maximize the number of dimer molecules by choosing the combination  $\kappa_1^+, \kappa_2^-, \kappa_3^+, \kappa_4^-, \kappa_5^+, \kappa_6^-, \kappa_7^-, \kappa_8^+, \kappa_9^-, \kappa_{10}^+$ . Note that although dimers are consumed by  $C_5$ , choosing  $\kappa_5^+$  maximizes the number of dimers in the system. This is because  $C_5$  is necessary to produce monomers and therefore also dimers.

We never run out of memory with the sliding window method, but the running times can be huge for a long time horizon. The reason is that the windows are large since the system contains many monomers and dimers at later time instances. For the results in Table 1 we considered the system till time  $t = 300$ . In [7], the longest time horizon is  $t = 100$ .

- Immune Response.** We consider a model for the intercellular interactions of an immune response evoked by virus infection [6]. Similar as for intracellular interactions, cell-cell interactions show intrinsic fluctuations due to process noise [28]. A state  $x = (x_T, x_I, x_E) \in \mathbb{N}_0^3$  is given by the number of target cells ( $T$ ), infected cells ( $I$ ), and effector cells ( $E$ ). The transition classes  $C_i = (G_i, u_i, \alpha_i)$ ,  $1 \leq i \leq 7$  are given as follows.

- Influx of new target cells:  $G_1 = \mathbb{N}_0^3$ ,  $u_1(x_T, x_I, x_E) = (x_T + 1, x_I, x_E)$ ,  $\alpha_1(x_T, x_I, x_E) = c_1$ .
- Infection:  $G_2 = \{(x_T, x_I, x_E) \mid x_T > 0, x_I > 0\}$ ,  $u_2(x_T, x_I, x_E) = (x_T - 1, x_I + 1, x_E)$ ,  $\alpha_2(x_T, x_I, x_E) = c_2 \cdot x_T \cdot x_I$ .
- Elimination of infected cells:  $G_3 = \{(x_T, x_I, x_E) \mid x_I > 0, x_E > 0\}$ ,  $u_3(x_T, x_I, x_E) = (x_T, x_I - 1, x_E)$ ,  $\alpha_3(x_T, x_I, x_E) = c_3 \cdot x_I \cdot x_E$ .
- Proliferation of effector cells:  $G_4 = G_3$ ,  $u_4(x_T, x_I, x_E) = (x_T, x_I, x_E + 1)$ ,  $\alpha_4(x_T, x_I, x_E) = c_4 \cdot x_I \cdot x_E$ .
- Death of target cells:  $G_5 = \{(x_T, x_I, x_E) \mid x_T > 0\}$ ,  $u_5(x_T, x_I, x_E) = (x_T - 1, x_I, x_E)$ ,  $\alpha_5(x_T, x_I, x_E) = c_5 \cdot x_T$ .
- Death of infected cells:  $G_6 = \{(x_T, x_I, x_E) \mid x_I > 0\}$ ,  $u_6(x_T, x_I, x_E) = (x_T, x_I - 1, x_E)$ ,  $\alpha_6(x_T, x_I, x_E) = c_6 \cdot x_I$ .
- Death of effector cells:  $G_7 = \{(x_T, x_I, x_E) \mid x_E > 0\}$ ,  $u_7(x_T, x_I, x_E) = (x_T, x_I, x_E - 1)$ ,  $\alpha_7(x_T, x_I, x_E) = c_7 \cdot x_E$ .

Here,  $c_1, \dots, c_7$  are positive reaction rate constants, which are chosen as follows [6]:  $c_1 = 100$ ,  $c_2 = 0.1$ ,  $c_3 = 1$ ,  $c_4 = 0.001$ ,  $c_5 = 0.1$ ,  $c_6 = 0.1$ ,  $c_7 = 0.01$ . The initial state of the system is  $y = (1000, 50, 5)$ .

With the following branches we obtain upper bounds on the state variables.

- For a maximal number of target cells, we choose  $\kappa_1^+, \kappa_2^-, \kappa_5^-$ . The remaining transition classes have no influence on the target cell population.
- We maximize the number of infected cells by choosing  $\kappa_1^+, \kappa_2^+, \kappa_3^-, \kappa_4^-, \kappa_5^-, \kappa_6^-, \kappa_7^+$ .
- We maximize the number of effector cells with  $\kappa_1^+, \kappa_2^+, \kappa_3^-, \kappa_4^+, \kappa_5^-, \kappa_6^-, \kappa_7^-$ .

The opposite branches minimize the respective populations.

Similar to Goutsias model, we never run out of memory though the windows are large. The results in Table 1 are for a time horizon of  $t = 3.5$  days.